Identifier des icebergs parmi des flux de données distribués

Emmanuelle Anceaume, Yann Busnel, Nicoló Rivetti, Bruno Sericola

Identifier des icebergs parmi des flux de données distribués

Emmanuelle Anceaume[1], Yann Busnel[2,3], Nicoló Rivetti[4,5] et Bruno Sericola[3]

[1]*IRISA / CNRS, Rennes, France*
[2]*Crest / Ensai, Rennes, France*
[3]*Inria Rennes – Bretagne Atlantique, France*
[4]*LINA / Université de Nantes, France*
[5]*Sapienza University of Rome, Italie*

Nous considérons dans cet article le problème d'identification d'attaques par déni de service distribué, dite "iceberg", dans des flux de données massifs, physiquement répartis. Une telle attaque consiste à cacher un déni de service parmi de multiple flux de données de façon à ce que le déni de service ne soit pas détectable sur chacun des flux pris en isolation. Une solution naturelle est pour chacun des routeurs de fournir régulièrement des informations à un coordinateur en charge de collecter et d'agréger celles-ci. Cependant pour que cette solution soit pertinente, la quantité d'informations transitant entre les routeurs et le coordinateur et la complexité en mémoire dédiée à l'analyse de chacun des flux doit être extrêmement faible par rapport au volume total des données reçues dans les différents flux. Dans cet article, nous proposons un algorithme probabiliste distribué efficace détectant les icebergs à la volée. Nous présentons brièvement l'analyse théorique des performances de notre algorithme (e.g. coût en espace mémoire, en nombre de message échangés et en temps de calcul) ainsi que les résultats expérimentaux obtenus sur un cluster de Raspberry Pi. Ces derniers confirment l'efficacité et la précision de notre algorithme pour identifier les icebergs cachés dans des flux de données massifs.

Mots-clefs : Flux de données distribués; Algorithme d'approximation probabiliste; Attaque par déni de service; Evaluation de performance.

Travaux co-financés par le projet ANR SocioPlug (ANR-13-INFR-0003) et le projet DeSceNt du Labex CominLabs (ANR-10-LABX-07-01).

1 Introduction

Une attaque par déni de service (DoS) tente de faire tomber une ressource en ligne en inondant cette ressource avec plus de requête qu'il n'est capable d'en traiter. Une attaque par déni de service distribuée (DDoS) possède le même objectif mais provoqué par plusieurs sources, préalablement infectées par un logiciel malveillant (malware), impliquant l'arrêt immédiat de la cible de l'attaque (tels que des sites de e-commerce par exemple). Appliquer une surveillance continue du trafic passant par les routeurs réseaux est une approche classique pour détecter et limiter ces attaques, via une analyse des signatures IP très fréquentes pouvant représenter des cibles potentielles. Cependant, une nouvelle forme de DDoS a récemment vu le jour, consistant a masquer les attaques en répartissant celles-ci stratégiquement sur différentes routes. Ainsi, localement, les sous-flux malveillants n'apparaissent pas comme une menace directe, mais globalement, ils représentent une proportion significative du trafic réseaux [MSDO05]. Le terme d'*attaque par iceberg* a été proposé pour représenter ce type d'attaque, chaque routeur ayant seulement la visibilité d'une infime partie de l'attaque [ZLOX10]. Une solution naturelle serait donc de centraliser les informations collectées localement sur chaque routeur par analyse de leur trafic respectif. Cela implique donc que *(i)* les routeurs soient capables de traiter en ligne un flux de très grande taille pour découvrir la présence potentielle d'une attaque par iceberg suffisamment tôt et *(ii)* les communications entre les routeurs et le coordinateur central soient réduites au maximum afin d'éviter que ce dernier soit lui-même la cible d'une attaque DDoS. Evidemment, diminuer la fréquence de ces échanges ne doit pas se faire au détriment de la latence de detection des icebergs et, en aucun cas, introduire de faux-négatifs (*i.e.*, non-détection d'un iceberg réel).

Dans cet article, nous proposons une solution au problème de détection et d'identification d'icebergs, tout en garantissant les deux propriétés ci-dessus. Afin de répondre à ce double objectif, l'utilisation de structure de données permettant de résumer des flux entrants parait incontournable [Mut05]. Celles-ci ont la capacité de maintenir à la volée une approximation fiable de fonctions f données sur un préfixe de flux d'entrée, potentiellement infini, et ce, en utilisant une quantité de mémoire très faible par rapport à la taille du flux considéré. De plus, ces algorithmes ne doivent pas être affectés par l'ordre dans lequel les éléments du flux sont reçus. En résumé, localement, seuls les menaces locales identifiables par un routeur sont surveillées et transmises au coordinateur, qui sera en charge d'interroger les autres routeurs pour vérifier la véracité de cette attaque. Plusieurs structures de données de faible taille sont utilisées pour *(i)* estimer la fréquence locale de chaque item du flux entrant, *(ii)* stocker temporairement les menaces potentielles et *(iii)* mémoriser les icebergs réellement identifiés sur le réseau. Nous montrons que notre solution est une (ε,δ)-approximation du problème d'identification des icebergs, pour tout $\varepsilon \in [0,1]$ et $\delta \leq 1/2$.

Ci-dessous, nous présentons le modèle du système et de l'attaquant, puis nous formalisons le problème d'identification d'icebergs et présentons notre solution. Enfin, nous introduisons les résultats issus de l'analyse formelle de celle-ci ainsi que ceux issus de l'implémentation du prototype sur un banc de test de plusieurs Raspberry Pi. Pour des raisons de contraintes d'espace, la présentation des travaux relatifs ainsi que les preuves de l'analyse théorique de notre solution ont été omises. Le lecteur intéressé les trouvera, ainsi que des résultats exhaustifs des expérimentations, dans la version intégrale de ces travaux [ABRS15].

2 Modèle du système et de l'adversaire

Pour analyser nos algorithmes, nous nous plaçons dans le modèle proposé par Cormode *et al.* [CMY08], qui combine les spécificités du modèle flux avec des aspects de complexité réseaux. En particulier, nous considérons un ensemble de S nœuds, recevant chacun un flux continu d'items, issus d'un univers $\mathcal{N}$ de grande taille. Les items arrivent rapidement, sans contraintes d'ordre ni d'unicité. Cela permet par exemple de modéliser la reception à haut debit d'une longue séquence de paquets TCP/IP sur chaque routeur considéré. Si dans notre exemple, les items correspondent aux adresses IP destinations des attaques, nous les noterons par des entiers $1,\ldots,n$, avec $n = |\mathcal{N}|$, par soucis de clarté. Chaque flux reçu par un nœud s ($s \in \mathcal{S}$) est noté σ_s et m_s représente le nombre d'item reçu sur σ_s jusqu'à lors. Les flux reçus respectivement sur deux nœuds distincts s et s' n'étant pas synchronisés, il est fort probable que m_s soit donc différent de m'_s. De plus, un flux d'entrée σ_s défini implicitement une distribution de probabilité empirique des items qu'il contient : la probabilité d'occurence $p_{j,s}$ d'un item j dans σ_s est approché par $f_{j,s}/m_s$, où $f_{j,s}$ représente le nombre d'occurrences de j dans σ_s reçu depuis le début du flux. Cette distribution n'est bien entendu pas connu des nœuds a priori. Afin de calculer des fonctions sur l'union de tous les flux, les nœuds communiquent uniquement avec un nœud dédié, dénommé *coordinateur*.

Enfin, nous considérons la présence d'un adversaire *adaptatif*, qui cherche a éviter la détection des icebergs en les répartissant judicieusement parmi les S flux. Il est ainsi capable d'insérer autant d'items que nécessaire pour augmenter ou réduire le nombre d'icebergs, par rapport à l'état courant du coordinateur et des S nœuds, ou de réordonner les flux à sa guise. Cependant, tous les nœuds suivent scrupuleusement les protocoles, lesquels sont des données publiques pour éviter toute sorte de sécurité par obscurité.

3 Problème de détection et d'identification d'icebergs

Le problème de détection (et d'identification) d'icebergs a été formalisé par Estan et Varghese [EV03], puis adapté dans différents contextes [MSDO05, ZLOX10]. Ici, nous étendons ce problème au cadre du modèle de surveillance fonctionnelle répartie [CMY08]. Informellement, ce problème consiste, pour le coordinateur, d'identifier rapidement tout item j excédant globalement une proportion donné du flux.

Formellement, nous noterons σ l'union des S flux σ_s depuis l'origine, soit $\sigma = \sigma_1 \cup \ldots \cup \sigma_S$. Dénotons alors par m le nombre d'items de σ. Nous avons donc $m = \sum_{s\in\mathcal{S}} m_s$. Enfin, f_j représentera le nombre total d'occurrence d'un item j dans σ.

Définition 3.1 (Problème de détection et d'identification d'icebergs) *Pour tout seuil $\Theta \in (0,1]$, tout paramètre d'approximation $\varepsilon \in [0,1]$, et toute probabilité d'erreur $\delta \leq 1/2$, le problème de détection et d'iden-*

tification d'icebergs consiste, pour le coordinateur C,

— *à retourner tout item j si $f_j \geq \Theta m$, et*

— *à ne jamais retourner un item j si $f_j < (1-\varepsilon)\Theta m$.*

4 Un algorithme d'identification fiable

Supposons dans un premier temps que pour tout item j reçu sur un flux σ_s, celui-ci est étiquetté par sa probabilité $p_{j,s}$. Chaque nœud $s \in \mathcal{S}$ reçoit donc un flux, mais ne garde trace que des éléments très fréquents, *i.e.*, ceux qui représentent localement une proportion au moins égale à Θ du flux σ_s (les autres étant simplement ignorés). Afin de traiter proportionnelement l'urgence des items extrêmement fréquents par rapport aux simplement fréquents, chaque nœud stocke ces items dans différents tampons. L'intervalle $[\Theta, 1]$ est découpé en ℓ sous-intervalles de taille géométrique, et chacun des ℓ tampons $\Gamma_k, k \in \{1, \ldots, \ell\}$ (et $|\Gamma_k| = c_k$), est associé aux items ayant une probabilité d'occurrence correspondant au $k^{\text{ème}}$ intervalle : $[\Theta + (1-\Theta)/2^k, \Theta + (1-\Theta)/2^{k-1}]$. Dès qu'un tampon est rempli, son contenu est alors envoyé au coordinateur. Ce dernier interroge ensuite les autres nœuds afin de déterminer si un iceberg est effectivement présent.

De plus, la distribution des items dans chaque flux σ_s étant inconnue, il est probable que ces tampons ne soient jamais remplis intégralement. Afin d'assurer que tout iceberg potentiel soit transmis au coordinateur, un minuteur τ_k est démarré dès qu'un item est ajouté à Γ_k, et incrémenté à chaque reception d'un nouvel item sur le flux σ_s. Le délai de temporisation de τ_k est initialisé à $H_{\lfloor 1/\Theta \rfloor}/\Theta$, où H_n représente le $n^{\text{ème}}$ harmonique défini par $H_0 = 0$ and $H_n = 1 + 1/2 + \cdots + 1/n$.

Enfin, à chaque détection d'un iceberg réel, le coordinateur en informe à la fois l'application et les S nœuds. Ces derniers l'ajoutent alors à une file spécifique Λ_s, de taille inférieure à $\lceil 1/\Theta \rceil$. Cela évite de retransmettre au coordinateur à maintes reprises un potentiel iceberg déjà détecté.

Revenons à présent sur l'hypothèse précédente. Connaitre la probabilité d'occurence d'un item j reçu sur σ_s est irréaliste. Cependant, nous pouvons l'estimer en temps réel, en utilisant une structure de données utilisant un faible espace mémoire, appelée *Count-Min (CM) Sketch* [CM05]. CM estime la fréquence $f_{j,s}$ de l'item j avec une erreur bornée par un facteur ε avec une probabilité δ. Cet estimateur utilise une matrice $\hat{F}_s$ de $s_1 \times s_2$ compteurs avec $s_1 = \lceil \log(1/\delta) \rceil$ et $s_2 = \lceil e/\varepsilon \rceil$, ainsi que des s_1 fonctions de hachage 2-universelles $h_1, \ldots, h_{s_1}$. A la reception de chaque item j sur le flux, un compteur pour chaque ligne est incrémenté, *i.e.*, $\hat{F}_s[v][h_v(j)]$ est incrémenté pour tout $v \in \{1, \ldots, s_1\}$. Lors d'une requête d'estimation de fréquence de j, le minimum des valeurs parmi $\hat{F}_s[v][h_v(j)]$ ($v \in \{1, \ldots, s_1\}$) est retourné. La taille mémoire requise par un CM Sketch est donc proportionnelle à $\frac{1}{\varepsilon} \log_2 \frac{1}{\delta}$ [CM05].

5 Analyse et évaluation de la solution

Nous montrons dans [ABRS15] que notre solution est une (ε, δ)-approximation permettant de résoudre le problème de la détection d'icebergs, pour tout $\varepsilon \in [0, 1]$ et $\delta \leq 1/2$, avec une faible complexité mémoire. De plus, nous fournissons une borne supérieure précise sur le nombre de bits échangés par notre solution. Ces résultats sont formalisés ci-dessous.

Dans un soucis de clarté, nous définissons $\tau(\Theta) = H_{\lfloor 1/\Theta \rfloor}/\Theta$, et ainsi, pour tout $k = 1, \ldots, \ell$, $\tau_k = \tau$. †

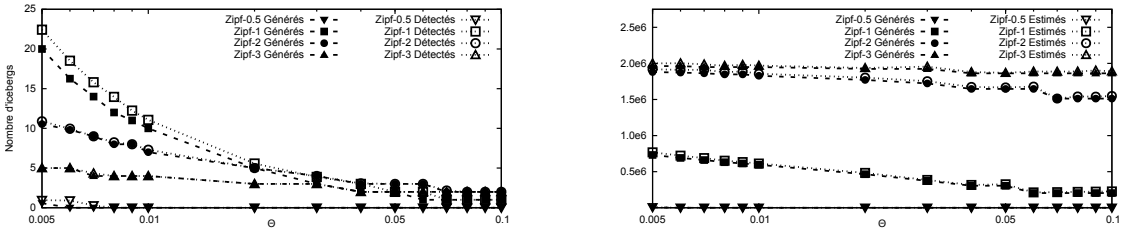
Théorème 5.1 ***(Borne supérieure du coût de communication)*** *La solution présentée dans la Section 4 et dans [ABRS15] ne transmet en moyenne pas plus de*

$$2mS(\log m + \log n) \sum_{k=1}^{\ell} \frac{c_k}{\sum_{i=0}^{\tau-1} \Pr\{T_{c_k,n_k}(v) > i\}} \text{ bits.} \tag{1}$$

Ici, la somme des probabilités au dénominateur de la relation 1 représente le temps d'atteinte moyen de l'emission d'un message, qu'il soit dû au remplissage du tampon k ou à la fin du délai de temporisation τ.

Théorème 5.2 ***(Correction et complexité en mémoire)*** *La solution présentée dans la Section 4 est une (ε, δ)-approximation résolvant le problème de détection d'icebergs (*cf. *Définition 3.1) et utilise $O((\log n + \log m)\log(1/\delta)(1/\Theta - 1)/\varepsilon + (\log S \log n)/\Theta)$ bits de mémoire sur chaque nœud [ABRS15].*

†. Afin de simplifier l'écriture, nous utiliserons la notation τ en lieu et place de $\tau(\Theta)$ s'il n'y a pas d'ambiguité.



(a) Nombre d'icebergs générés et détectés.

(b) Fréquence gobale des icebergs (générées et estimées).

FIGURE 2: Efficacité et précision de notre solution, en fonction de Θ, où $m = 2\,000\,000$ et $n = 10\,000$.

Enfin, nous avons implémenté notre solution sur un prototype déployé sur un banc de test composé de 20 Raspberry Pi Modèle B et deux serveurs interconnectés par un réseau Ethernet Gigabit. Chaque Raspberry héberge un nœud, l'un des deux serveurs joue le role du coordinateur et l'autre génère les S flux à destination des nœuds. Les flux ont été produits à la fois à partir de trace réelles mais également générés synthétiquement, afin de mettre en exergue des comportements difficiles à obtenir sur des traces existantes, démontrant ainsi la robustesse de notre solution. Nous ne pouvons présenter dans cet article qu'un aperçu des résultats obtenus : précision et rappel de notre solution (Figures 1 et 2(a)), et estimation de fréquence globale (nombre d'occurrences globales) des icebergs (Figures 2(b)). Par précision, nous définissons le nombre global d'icebergs réels détectés par notre solution divisé par le nombre total d'items détectés comme tels. Par rappel, nous définissons le nombre total d'icebergs réels détectés par notre solution divisé par le nombre total d'icebergs réellement présents. Les résultats numériques confirment clairement que pour toute valeur de Θ, le rappel est toujours égal à 1 : tous les icebergs ont été détectés (*i.e.*, pas de faux négatifs). Les Figures 1 et 2(a) illustrent également que la précision est très haute pour des distributions Zipf de paramètre $\alpha \geq 1$, mais diminue ensuite. En effet, pour des petites valeurs de α, il existe peu d'items fréquents, et la différence de fréquence entre des véritables icebergs et ceux dont la probabilité d'occurence est inférieurement proche à Θ est toute petite (de l'ordre de $1,5 \times 10^{-3}$ pour une valeur de $\Theta = 5 \times 10^{-3}$). D'autres résultats numériques montrent que le coût de communication généré par notre solution reste très faible (moins de 1.2% du nombre global d'items reçus par les routeurs). Enfin, ils illustrent que moins de 3% de lecture du flux global est nécessaire pour détecter tous les icebergs, sans variation majeure de la distribution de ceux-ci.

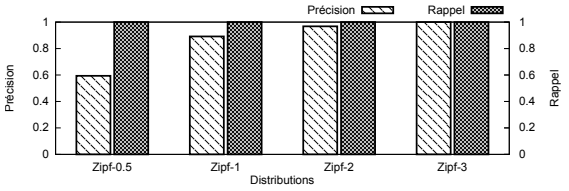


FIGURE 1: Precision et rappel en fonction du type de flux entrant pour $\Theta = 0.005$, où $m = 2\,000\,000$ et $n = 10\,000$.

Références

[ABRS15] E. Anceaume, Y. Busnel, N. Rivetti, and B. Sericola. Identifying global icebergs in distributed streams. In *Proc. of the 35th IEEE Symposium on Reliable Distributed Systems (SRDS'15)*, Montreal, Canada, 2015.

[CM05] G. Cormode and S. Muthukrishnan. An improved data stream summary : the count-min sketch and its applications. *Journal of Algorithms*, 55(1) :58–75, 2005.

[CMY08] G. Cormode, S. Muthukrishnan, and K. Yi. Algorithms for distributed functional monitoring. In *Proc. of the 19th annual ACM-SIAM Symposium On Discrete Algorithms (SODA)*, 2008.

[EV03] C. Estand and G. Varghese. New directions in traffic measurement and accounting : Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3) :270–313, 2003.

[MSDO05] A. Manjhi, V. Shkapenyuk, K. Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *Proc. of the 21st IEEE International Conference on Data Engineering (ICDE)*, 2005.

[Mut05] S. Muthukrishnan. *Data Streams : Algorithms and Applications*. Now Publishers Inc., 2005.

[ZLOX10] Q. Zhao, A. Lall, M. Ogihara, and J. Xu. Global iceberg detection over distributed streams. In *Proc. of the 26th IEEE International Conference on Data Engineering (ICDE)*, 2010.